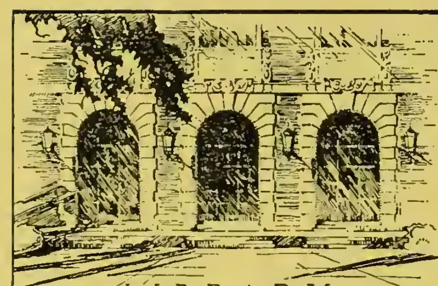


Report No. 2[illegible]

Math

COMPUTATIONAL EXPERIENCE IN ALL-INTEGER, BINARY VARIABLE, INTEGER PROGRAMMING PROBLEMS USING GOMORY'S ALL-INTEGER ALGORITHM

by

Charles R. Baugh
Toshihide Ibaraki
Saburo Muroga

April 4, 1968

DEPARTMENT OF COMPUTER SCIENCE · UNIVERSITY OF ILLINOIS · URBANA, ILLINOIS

Report No. 259

COMPUTATIONAL EXPERIENCE IN ALL-INTEGER, BINARY VARIABLE, INTEGER PROGRAMMING PROBLEMS USING GOMORY'S ALL-INTEGER ALGORITHM

by

Charles R. Baugh
Toshihide Ibaraki
Saburo Muroga

April 4, 1968

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

ABSTRACT

Our computational experience with Gomory's algorithm for the integer linear programming problem of synthesis of optimum network with NOR gates is presented. The problem is briefly described and accompanied with statistics such as the size and density of the coefficient matrix.

Upon successful solution of problems with 90 variables and 240 rows, the effect of constraint orderings and adding additional inequalities was investigated. The difference in convergence of two of Gomory's pivot selection rules is noted. Also the behavior in convergence of feasible versus non-feasible problems is demonstrated.

The convergence rate is conjectured to be $A \cdot 10^{Bn}$ where n is the number of variables of a switching function to be synthesized and A, B are constant coefficients. This rate is compared to an exhaustive method developed by Hellerman to solve the same logical design problem.

TABLE OF CONTENTS

1. INTRODUCTION

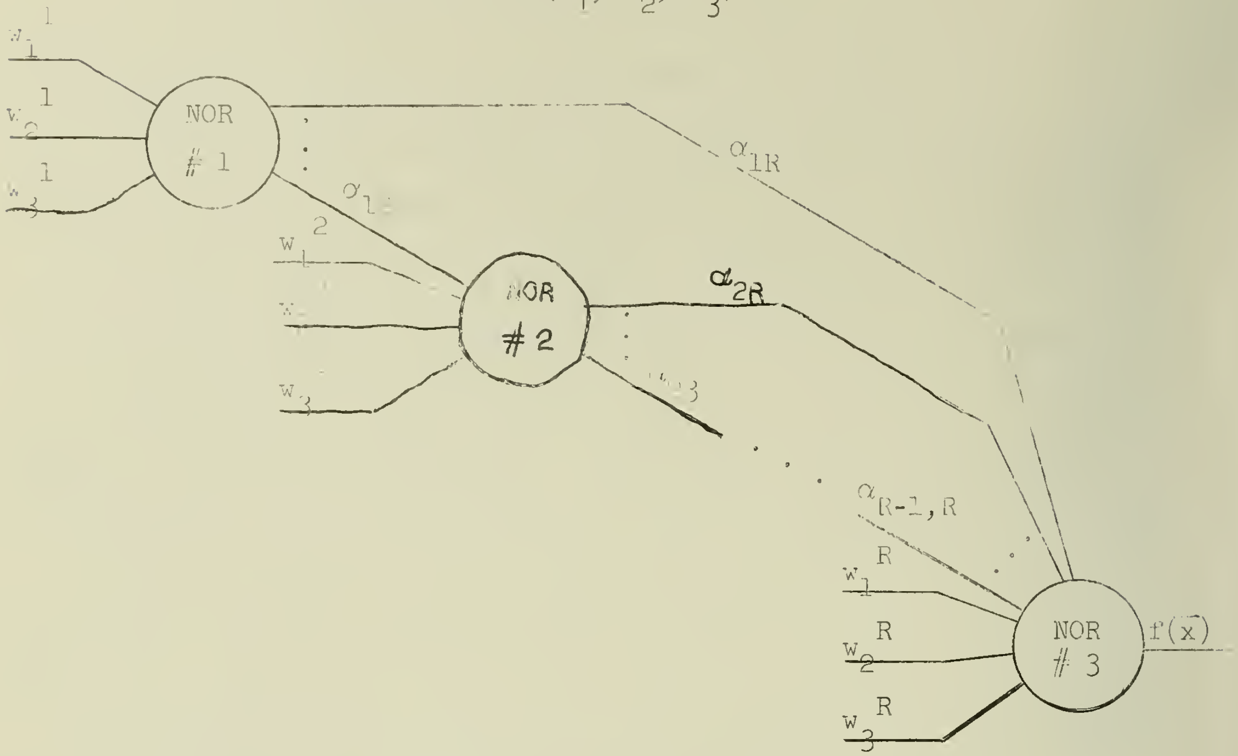
Our computational experiments were performed using Gomory's all-integer integer linear programming method even though our variables are restricted to be 1 or 0. The problems are unusual in that the number of inequalities is larger than the number of variables. This is just the opposite of the vast majority of the other published computational reports. Also the size of the coefficient matrix is much larger - up to 90 variables and 240 constraints. The problem formulation is derived from the design of logic circuits in digital computers. Specifically it is concerned with the optimum synthesis of a NOR element network complicated by considering fan-in and fan-out restrictions. In the next section the set of inequalities will be stated and briefly explained.

Previous publications have demonstrated the very erratic behavior in convergence. Some problems have been solved in only a few iterations while others needed several hundred thousand iterations for a solution.[(9),(10)] The convergence rate of Gomory's method seems to be highly dependent upon the characteristic of the constraints of each particular problem.

We will examine the rate of convergence and the effect that certain factors have upon it. Some of the factors we will consider are the order of the inequalities, the addition of other constraints, and the difference between feasible and non-feasible solutions.

2. PROBLEM STATEMENT

Fig. 1: Feed Forward NOR Circuit For the Boolean Function $f(x_1, x_2, x_3)$



Our integer problem arises from the attempt to synthesize a Boolean function f of three variables, x_1, x_2, and x_3 with a feed forward NOR element circuit as is shown in figure 1.* In the figure we denote the weight of an input variable x_i to the j-th NOR element by w_i^j and the weight of the output from the k-th NOR element to the e-th element by α_{ke}. The logical design problem is to determine which α's and w's are 1 (connected) and which are zero (disconnected) by using Gomory's all-integer algorithm. It is possible that the given number of elements R is insufficient for a particular function, i.e. the problem is infeasible. Therefore we may have a considerable number of infeasible problems if R is small.

---

* For a detail description of the general feed forward network synthesis formulation of which this problem is a special case see reference (8).

In order to model the circuit with linear inequalities many other variables must be appended to the original w's and α's. These constraints for the R element feed forward network are written in terms of the integers α's, w's, P's, and $\alpha^{(j)}$'s as follows*:

For the k-th element $\quad (k = 1, 2, \ldots, R-1)$

$$\underline{A_k}: \quad U \geq \sum_{i=1}^{3} w_i^{k} x_i^{(j)} + U P_k^{(j)} + \sum_{i=1}^{k-1} (\alpha_{ik} - \alpha_{ik}^{(j)})$$

$$\underline{B_k}: \quad -1 \geq -\sum_{i=1}^{3} w_i^{k} x_i^{(j)} - U P_k^{(j)} - \sum_{i=1}^{k-1} (\alpha_{ik} - \alpha_{ik}^{(j)})$$

$$\underline{C_k}: \quad 0 \geq \alpha_{ek} - \alpha_{ek}^{(j)} - P_e^{(j)}$$

$$0 \geq -\alpha_{ek} - \alpha_{ek}^{(j)}$$

$$1 \geq P_e^{(j)} + \alpha_{ek}^{(j)} \qquad \text{for } e = 1, \ldots, k-1$$

$$\underline{D_k}: \quad 3 \geq \sum_{i=1}^{3} w_i^{k} + \sum_{i=1}^{k-1} \alpha_{ik}$$

$$3 \geq \sum_{i=k+1}^{R} \alpha_{ki} \qquad \text{for } j = 1, 2, \ldots, 8$$

For the Rth element

$$E: \quad 0 \geq \sum_{i=1}^{3} w_i^{R} x_i^{(j)} + \sum_{i=1}^{R-1} (\alpha_{iR} - \alpha_{iR}^{(j)}) \qquad \text{for } f(x_1^{(j)}, x_2^{(j)}, x_3^{(j)}) = 1$$

$$-1 \geq -\sum_{i=1}^{3} w_i^{R} x_i^{(j)} - \sum_{i=1}^{R-1} (\alpha_{iR} - \alpha_{iR}^{(j)}) \qquad \text{for } f(x_1^{(j)}, x_2^{(j)}, x_3^{(j)}) = 0$$

$$0 \geq \alpha_{eR} - \alpha_{eR}^{(j)} - P_e^{(j)}$$

$$0 \geq \alpha_{eR} - \alpha_{eR}^{(j)}$$

$$1 \geq P_e^{(j)} + \alpha_{eR}^{(j)} \qquad \text{for } e = 1, 2, \ldots, R-1$$

---

* For a detailed derivation of these equations see reference (8).

$\underline{F}$: $$3 \geq \sum_{i=1}^{3} w_i^{R} + \sum_{i=1}^{R} \alpha_{iR} \qquad \text{for } j = 1, 2, \ldots, 8$$

For all R elements

$\underline{G}$: all w^k's, α_{ij}'s, $\alpha_{ke}^{(j)}$'s, and $P_i^{(j)}$'s ≤ 1.

These seven groups of constraints A-F compose the complete set of inequalities. The constant U is a sufficiently large positive value so that if $P_k^{(j)} = 0$ the j-th inequality of A_k is always satisfied and if $P_k^{(j)} = 1$ the j-th inequality of B_k is always satisfied. The input vector $\vec{x}^{(j)} = (x_1^{(j)}, x_2^{(j)}, x_3^{(j)})$ for the three variable Boolean function $f(\vec{x})$ assumes all eight possible combinations of 1 and 0. Inequalities D_k and F make the further restrictions that no NOR element can have more than 3 inputs or more than 3 outputs. The fan-in, fan-out constraints (inequalities D_k and F) arise from practical engineering restrictions on the actual electronic circuits which are used to realize the NOR elements.

Therefore our all-integer integer linear programming problem is:

$$\min \; g = \sum_{k}^{R} \sum_{i=1}^{3} w_i^{k} + \sum_{e=1}^{R-1} \sum_{k=e+1}^{R} \alpha_{ek}$$

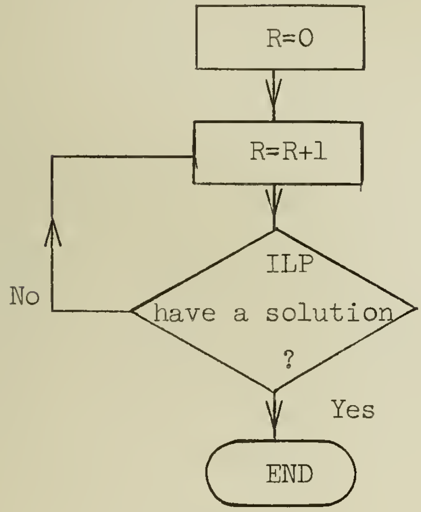
under the constraints

A through G

By minimizing g we are minimizing the total number of connections. The procedure for determining a network for a given Boolean function is outlined in figure 2.

Fig. 2: Flow Chart Of Procedure

of Synthesizing A Boolean Function



In this manner we will obtain a network with the fewest number of NOR elements and with minimum connections.

The number of variables, inequalities, entries, and non-zero entries in the coefficient matrix all increase with the number of elements R. Figure 3 shows this growth. Slack variables are not included.

Fig. 3: Characteristics Of The Coefficient Matrix

R	Matrix Size		Coefficients		
	Constraints	Variables	Total No.	Non-zero*	%Non-zero
1	11	3	33	18	55.50
2	64	23	1472	183	12.40
3	142	52	7380	447	6.08
4	245	90	22,040	810	3.68
5	374	137	51,250	1277	2.49
6	528	193	101,800	1836	1.81
7	707	258	182,200	2487	1.37

* This number is for the Boolean function $f(\vec{x}) \equiv 1$. See the E inequalities for the effect of other Boolean functions.

3. OUTLINE OF RESULTS

The algorithm was programmed in assembly language NICAP for the ILLIAC II computer at the University of Illinois. The computer has a 1.75 microsecond core cycle time, an 8K - 52 bit word memory, and the facility to operate on 13 bit quarter words. All the coefficient matrix entries were stored in the 13 bit quarter words. Therefore whenever the algorithm generates a number greater in absolute value than 4096, the computation terminates with overflow.

Gomory in proposing his all-integer integer program suggested several variations to the basic algorithm.[1] We used the variation which chooses the pivot row which minimizes the lexiographic rank of the pivot column. In order to ascertain the factors important in convergence of the algorithm, we solved several problems. These same problems were then reformulated by changing the order of inequalities, by adding additional constraints, by changing the value of U, etc. In our tests the number of problems run was not large enough to guarantee statistical accuracy. This is especially true since convergence is very unpredictable. However the results do demonstrate the qualitative tendency of integer programming.

The integer linear programming formulation with R = M models a feed-forward network of M NOR elements If a Boolean function is realizable with M or fewer elements, the corresponding integer linear program has an optimal feasible solution. But if a function requires more than M elements, the problem is not feasible.

The set of Boolean functions upon which we performed our experiments is shown in figure 4. Hellerman's[12] network numbering system was used to uniquely identify each function and will be used throughout this report.

Fig. 4: Complete Set of Boolean Test Functions

Boolean Function	Hellerman network number	Number of gates in network	Objective function value
$\bar{a}$	4D	1	1
$a\bar{b}$	5D	1	2
$a \vee b$	6D	2	3
$\bar{a}b$	7D	2	3
$\bar{a} \vee b$	8D	3	4
ab	9D	3	4
$\bar{a}\bar{b}\bar{c}$	1	1	3
$a \vee b \vee c$	2	2	4
$\bar{a}b \vee \bar{a}c$	3	2	4
$\bar{a}\bar{b}c$	4	2	4
$\bar{a}\bar{b} \vee c$	5	3	5
$\bar{a} \vee b \vee c$	6	3	5
$\bar{a}\bar{b} \vee \bar{a}c$	7	3	5
$ac \vee bc$	8	3	5
$ab \vee c$	9	3	6
$\bar{a}b \vee c$	10	3	5
$\bar{a} \vee \bar{b}$	10D	4	5
$ab \vee \bar{a}\bar{b}$	11D	4	8
$\bar{a}b \vee c$	11	4	6
$\bar{a} \vee \bar{b} \vee c$	12	4	6
$\bar{a} \vee \bar{b}\bar{c}$	13	4	6
$\bar{a}\bar{b} \vee \bar{a}\bar{c}$	14	4	6
$\bar{a}c \vee bc$	15	4	6
$\bar{a}b \vee ac$	16	4	7
abc	17	4	6
$ab \vee ac \vee bc$	18	4	9
$\bar{a} \vee bc$	19	4	7
$\bar{a}\bar{b} \vee bc$	20	4	8
$\bar{a}bc \vee \bar{a}\bar{b}c$	21	4	9
$\bar{a}\bar{b} \vee bc \vee ac$	22	4	9
$\bar{a}\bar{b}\bar{c} \vee bc$	23	4	9
$\bar{a}\bar{b} \vee ab \vee c$	24	4	10
$\bar{a}\bar{b}\bar{c} \vee ac \vee bc$	25	4	10

The density of non-zero coefficients in the coefficient matrix after each iteration is also of interest. The density of the initial matrix is low (see figure 3). In a seemingly typical problem the density starts at 5% and increases gradually until it reaches about 15%. It oscillates between 10% and 20% until convergence at which time it drops down to about 10%.

First we noticed the size of U in equations A and B considerably effects the occurrance of overflow. Generally the larger the value of U the more often overflow was encountered. By reducing the value of U from 9 down to 4 we were able to prevent overflow from occurring as early. This is especially true for problems which require a large number of iterations.

Gomory also suggested another approach[1] which he called the row combination method. Using this method on a set of Boolean functions for R = 3. The comparison between the minimum rank method and the minimum rank method appended with the row combination technique is seen in figure 5.

Fig. 5: Evaluation of Adding Row Combination

	Row combination	Without row combination
Ave. number of iter.	62.1	65.6
Time per iteration	465 msecs	166 msecs
% of problems with overflow	17.6%	0.0%

Thus in considering the increased computation for each iteration, the row combining technique should be disregarded.

4. FEASIBLE AND NON-FEASIBLE

For example, when a Boolean function realizable with R = 4 (4 NOR elements) is tried with the R = 3 formulation, the resulting integer linear program is not feasible. In Figure 6 the average number of iterations for all Boolean test functions and for three different orderings of the constraints are shown.*

Fig. 6: Average No. of Iterations For 3 Different Orderings of the Inequalities.*

	1	2	3
Feasible	72.5	66.0	66.6
Non-feasible	85.5	83.3	98.2

The difference in the average number of iterations for any case is not too great. However in the non-feasible cases the number of iterations deviates from the average much more than in the feasible case. For example in the ordering of column 3 of figure 6 all non-feasible functions required less than 92 iterations except for #17 and #18 of figure 4 which require 461 and 296 iterations respectively. The convergence varies over a much larger number of iterations in non-feasible problems than in feasible problems.

Figure 6 also demonstrates that non feasible problems are much more difficult to solve than the feasible problems.

---

* See figure 7 for details of the constraint ordering.

5. ORDER OF INEQUALITIES

Gomory's method with the incorporation of the lexiographic row rank variation[(1)] is sensitive to the order of inequalities since the lexiographic ordering is changed by altering the order of the constraints. Figure 7 displays the average number of iterations for all the functions of figure 4 for four different constraint orderings. The objective function is denoted by O.F. and slack variables always

Fig. 7: Average Number of Iterations
Under 4 Different Constraint Orderings

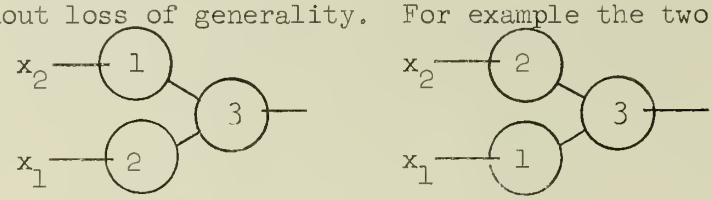
	1	2	3	4
	O.F.	O.F.	O.F.	O.F.
	G	E	E	A_1
	E	B_1	G	B_1
	B_1	B_2	B_1	A_2
	B_2	D_2	B_2	B_2
	D_2	F	D_2	C_2
	F	A_3	F	E
	A_1	C_3	A_3	A_3
	A_2	A_2	C_3	C_3
	C_2	C_2	A_2	G
	A_3	A_1	C_2	D_2
	C_3	G	A_1	F_2
Feasible	72.5	66.0	66.6	50.0
Non-feasible	85.5	83.3	98.2	118.0

appear after all other constraints. It should be noted that B_1, B_2, and E are the only constraints which initially have negative entries in the constant column.

It is difficult to draw any conclusive results from figure 7. However, it seems best to place the E constraints first since we have more non-feasible than feasible problems. These inequalities are the only ones which change according to the Boolean function which is being tested. Since the rest of the constraints remain the same regardless of the Boolean function being solved, the E constraints are likely the most important. Therefore we place them at the top. One of the reasonable orderings may be to put the more important inequalities at the upper part of the tableau. However, strictly speaking we don't have any measure of the importance of an inequality. One intuitive and reasonably successful measure is to count the number of non-zero entries in an inequality - the greater the number of coefficients the more important the constraint.

6. ADDITIONAL CONSTRAINTS

Another technique for increasing the speed of convergence is to incorporate additional constraints which exclude unnecessary solutions without loss of generality. For example the two networks;



realize the same Boolean function. But it is not important to get both solutions. Either one will be acceptable. Therefore additional constraints should be added to exclude either one but obviously not both.

By considering other properties of a 4 element NOR network ($R = 4$) we establish the following constraints:

$$
\begin{array}{lr}
 & \alpha_{24}-\alpha_{34} \leq 0 \\
 & \alpha_{14}-\alpha_{24} \leq 0 \\
 & \alpha_{23}+\alpha_{24} \leq 1 \\
 & \alpha_{24}-\alpha_{34} \leq -1 \\
 & -\alpha_{34} \leq -1 \\
 & \alpha_{12}+\alpha_{13}+\alpha_{14} \leq 2 \\
 & \alpha_{12}+\alpha_{13} \quad -\alpha_{24} \leq 1 \\
 & \alpha_{12}-\alpha_{13} \quad +\alpha_{24} \leq 1 \\
 & \alpha_{12}+\alpha_{13}+\alpha_{23}-\alpha_{24} \leq 2 \\
-(w_1^{\,1}+w_2^{\,1}+w_3^{\,1}) & +\alpha_{12}) \leq -1 \\
\quad -(w_1^{\,2}+w_2^{\,2}+w_3^{\,2} & +\alpha_{12} \leq -1 \\
\qquad -(w_1^{\,3}+w_2^{\,3}+w_3^{\,3} & +\alpha_{13}+\alpha_{23}) \leq -1 \\
\qquad\quad -(w_1^{\,4}+w_2^{\,4}+w_3^{\,4} & +\alpha_{24}+\alpha_{34}) \leq -1 \\
 & -(\alpha_{12}+\alpha_{13}+\alpha_{14}) \leq -1 \\
 & (\alpha_{23}+\alpha_{24}) \leq -1
\end{array}
$$

Note that some of the inequalities require that the network consist of exactly 4 elements while others require that the Boolean function have exactly 3 input variables. The additional constraints for the R = 3 network are obtained by neglecting the 1st element and relabeling the remaining elements 2, 3 and 4 as 1, 2, and 3 respectively.

Figure 8 shows the effect of adding these constraints to the first 3 row orderings of figure 7 for the R = 3 network and for the six Boolean functions #5-#10. Figure 9 demonstrates the improvement on the R = 4 network for all the 4 gate realizable Boolean functions. Without the additional constraints all the R = 4 network problems either did not converge after about 1200 iterations or generated an overflow. Although all the infeasible R = 4 problems tested with additional constraints did not converge after 1200 iterations, the additional constraints were extremely successful for feasible problems in the R = 4 case.

Fig. 8: Effectiveness of Additional Constraints on R = 3 Network. Average Number of Iterations

	Row Ordering					
	1		2		3	
	without	with	without	with	without	with
Feasible	87.5	68.2	103.7	70.1	87.7	68.1
Non-feasible	98.2	129.8	145.8	137.4	83.3	146.2

Fig. 9: Convergence with Additional Constraints on R = 4 Network

	without	with
% problems with no convergence	*	35.3%
% problems with overflow		11.8%
Average # iter. for sol.		355.6

From the results we notice the fact that the additional constraints facilitate convergence if the problem is feasible but make convergence worse if it is non-feasible. Furthermore, the incorporation of the added inequalities is much more effective with the R = 4 problems than the R = 3 problems.

* A few functions were tried. However all exceeded the preassigned bound of 1200 iterations.

7. FIXING THE VALUES OF A VARIABLE

We tried splitting the given problem into two smaller problems by fixing a particular variable to 1 in one and to 0 in the other. By comparing the results, we can pick the optimal network. By picking α_{13} in the R = 3 network and by using the formulation of column 4 of figure 7 we obtained the results of figure 10.

Fig. 10: Average Number of Iterations For All Test Functions

	α_{13} not preset	$\alpha_{13} = 0$	$\alpha_{13} = 1$
Feasible	71.0	20.0	18.5
Non-feasible	81.0	32.9	55.5

Generally it is difficult to judge whether fixing a variable speeds up the computation. However, it may be worthwhile if the right choice of a variable is made.

8. RELATIONSHIP BETWEEN THE NUMBER OF ITERATIONS AND VARIABLES

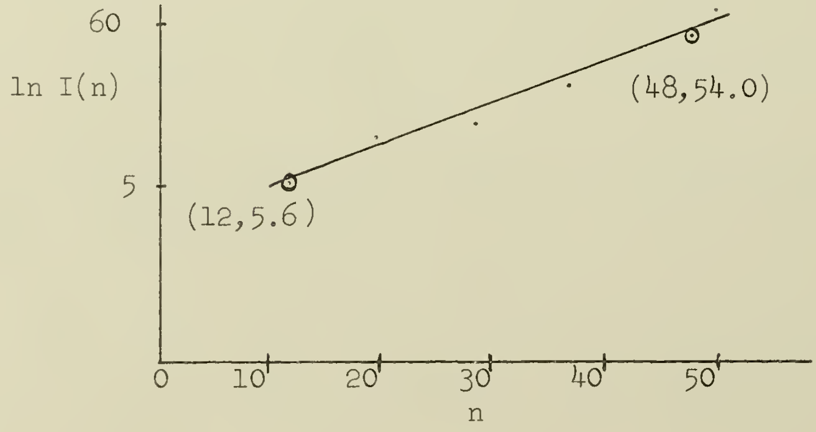
In order to determine the increase in the number of iterations we preset some randomly picked variables to their solution values. (Solution of our problem is already known by using another approach.)

This equivalently shrinks the size of integer linear programming problems, but the general characteristics of the problem may not change very much. The test was performed on one particular Boolean function ac $\vee$ bc with the following order of inequalities

OF
A_1
B_1
A_2
B_2
C_2
E
C_3
F
G

Each size of the coefficient matrix was run 5 times with a different set of variables being preset. The average number of iterations is plotted in figure 11.

Fig. 11: Number of Iterations I(n) as a Function of the Number of Variables n For the Boolean Function ac $\vee$ bc.



The curve of figure 11 demonstrates that the increase in the number of iterations grows exponentially and can be expressed as

$$I(n) \cong 2.5 \times 10^{.0277n}$$

for the Boolean function ac $\vee$ bc.

Our conjecture is that the number of iterations is generally $A \cdot 10^{Bn}$ where A and B are constants which depend upon the particular type of integer linear programming problem.

9. COMPARISON OF HELLERMAN'S ALGORITHM AND GOMORY'S

Helllerman in reference 12 determined the optimum NOR circuit for all Boolean functions $f(x_1, x_2, x_3)$ by an exhaustive method. He generated all possible circuits and chose the best one for each function.

It is interesting to compare the integer programming approach using Gomory's method with Hellerman's to determine if it is better. For the 3 and 4 element formulation ($R = 3, 4$) the computational efficiency of Gomory's method is inferior to Hellerman's approach. The number of iterations in Hellerman's method increases as $T \cdot 10^{CR^2}$ with the number of elements R.

In the previous section we showed that Gomory's method apparently increases as $A \cdot 10^{Bn}$. Therefore it is unlikely that Gomory's algorithm would be better than Hellerman's exhaustive method since n increases as R^2. However, this cannot be a definite conclusion since we do not have sufficient computational experience. Also some techniques to reduce the number of iterations for Gomory's method may be developed in the future which could make it more effective than Hellerman's approach.

10. CONCLUSIONS

Using Gomory's algorithm we successfully solved a few rather large 0 - 1 value all integer problems of 90 variables and 240 constraints.

Gomory's work was followed by the implicit enumeration methods of Balas[2], Geoffrion[4,11] and others[3,5,6,7]. These algorithms further restrict the problem by requiring all variables to be 1 or 0. The number of iterations for implicit enumerations reportedly grow as n^k.[11]

Since our problems are 0 - 1 problems, we are now experimenting with these implicit enumeration methods. We are very encouraged with the initial results and feel these methods may be better for our problems. Our results will be published.

(1) R.E. Gomory, "An all-integer integer programming algorithm", Industrial Scheduling edited by J.R. Muth and G.L. Thompson, Prentice-Hall, 1963.

(2) E. Balas, "An additive algorithm for solving linear programs with zero-one variables", Operations Research, vol. 13, No. 4, pp 517-544, July - August, 1965.

(3) F. Glover, "A multiple phase-dual algorithm for the zero-one integer programming problem", Operations Research, vol. 13, No. 6, pp 879-919, November - December, 1965.

(4) A.M. Geoffrion, "Integer programming by implicit enumeration and Balas' method", SIAM Review, vol. 9, No. 2, pp 178-190, April, 1967.

(5) R.J. Freeman, "Computational experience with a 'Balasian' integer programming algorithm", Operations Research, vol. 14, No. 5, pp 935-941, September - October, 1966.

(6) B. Fleisehmann, "Computational experience with the algorithm of Balas", Operations Research, vol. 15, No. 1, pp 153-155, January - February, 1967.

(7) C.C. Petersen, "Computational experience with variants of the Balas algorithm applied to the selection of R and D projects", Management Science, vol. 13, No. 9, pp 736-745, May, 1967.

(8) S. Muroga, "Threshold logic", lecture notes for EE 497 and EE 498, Department of Computer Science, University of Illinois, 1965-1966.

(9) J. Haldi and L.M. Isaacson, "A computer code for integer solutions to linear programs", Operations Research, vol. 13, No. 6, pp 946 - 959, November - December, 1965.

(10) G.T. Martin, "Directed network models for discrete programming", Paper presented at the International Symposium on Mathematical Programming, London School of Economics, July 1964.

(11) A.M. Geoffrion, "Recent computational experience with three classes of integer linear programs", Rand Report P-3699, October, 1967.

(12) L. Hellerman, "A catalog of three-variable OR-invert and AND-invert logical circuits", IEEE Transactions on Electronic Computers, vol. EC-12, pp. 198-223, June 1963.

(13) F.T. Chen, "Code for integer linear programming problem for Gomory's algorithm II", Department of Computer Science, University of Illinois, 1968.